UNIVERSITY OF
CAMBRIDGE

Comparing Machine Learning Techniques for Mobility Graph Classication

Miteyan Patel

Robinson College

Submitted on 20th May, 2018

Abstract

Declaration

I, Miteyan Patel of Robinson College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed:

Date:

Miteyan Patel

December, 2017

Acknowledgements

Contents

Chapter 1

Introduction

Machine learning techniques typically use structured and organised data which may not always be easily available in every day life. I will be investigating the use of unstructured graphical data as input into the most popular machine learning algorithms to evaluate the practicality of this and also apply new cutting edge algorithms to determine if we can classify users based on graphs created from their smartphone location data to infer their demographic class as labels as a supervised learning problem. Then I will create a mobile application to collect then cluster the location data into graphs for use by the most successful of the supervised algorithms investigated. If successful, this can have the great implications towards the utilisation of the data we have and can use in the future to create better classifiers and for example this can be used as a framework into using location data to infer whether or not users have neurological diseases such as Alzheimers which have known to alter the movement patterns of its carriers.

Chapter 2

Preparation

My preparation consisted of background reading and thoroughly understanding various supervised machine learning algorithms so that I could optimize them, as well as studying common techniques in conducting a machine learning project such as evaluation of the algorithms. Also researching graph theory and finding out features of graphs that could be extracted from the locations of the users to be used an input into the supervised learning algorithms.

2.1 Supervised machine learning

Supervised learning algorithms are used to infer the mapping function from labeled training data. Which consists of feature vectors and an associated label. Input feature vectors $\mathbf{x} \in \mathbb{R}^m$ are associated with a label y. In classification problems, y represents one of the possible output classes $\mathbf{C} = \{C_1, ..., C_k\}$. A training set is composed of n such training examples: $\mathbf{s} = [(\mathbf{x_1}, y_1)(\mathbf{x_2}, y_2)...(\mathbf{x_n}, y_n)]$ The goal is to approximate the mapping function called the hypothesis $h : \mathbb{R}^m \rightarrow \mathbf{C}$, so well that when you have new input data $\mathbf{x}$ that you can predict the output class for that data.

2.1.1 Neural Networks

A type of supervised learning algorithm commonly used is the Neural Network. It defines some function $f(\mathbf{x}; \mathbf{w}, \mathbf{b})$, dependent on its architecture to approximate the hypothesis, where $\mathbf{x}$ is the input vector, and $\mathbf{w}$ and $\mathbf{b}$ are the weights and biases in the network respectively. The Neural Network learns over successive iterations of the training algorithm the values of weights and biases by minimising a loss function, which defines to measure the models error when estimating y from $\mathbf{x}$, given initially random choices for $\mathbf{w}$ and $\mathbf{b}$.

2.1.2 Naive Bayes Classifier

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with naive independence assumptions between the features. This is the major problem with this approach since the features derived from the graphs are likely dependent on each other, although an advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. The Bayes theorem states describes the likelihood of an event A given an event B is true probability, based on prior knowledge of conditions that might be related to the event.

$$P(C_k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_k)\, P(C_k)}{P(\mathbf{x})}$$

In reality only the numerator is of interest since the denominator does not depend on the class **C** so it is effectively a constant and can be taken out and rewritten as:

$$\begin{aligned} P(C_k, x_1, x_2, ..., x_n) &= P(x_1 \mid C_k, x_2, ..., x_n) P(x_2, ..., x_n, C_k) \\ &= P(x_1 \mid C_k, x_2, ..., x_n) P(x_2 \mid x_3, ..., x_n, C_k) P(x_3, ..., x_n, C_k) \\ &= ... \\ &= P(x_1 \mid C_k, x_2, ..., x_n) ... P(x_n - 1 \mid x_n, C_k) P(x_n \mid x_n, C_k) P(C_k) \end{aligned} \tag{2.1}$$

Now with the naive assumption that each feature x_i is conditionally independent of one another so:

$$P(x_i \mid x_i + 1, ..., x_n, C_k) = P(x_i \mid C_k) \tag{2.2}$$

Therefore the conditional probability of the class variable C is:

$$P(C_k, x_1, x_2, ..., x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.3}$$

Finally to construct the naive Bayes classifier the most probable hypothesis is chosen known as the maximum a posteriori decision rule and the corresponding Bayes classifier is the function that assigns a class label $\hat{y} = C_k$, for some class C_k as:

$$\hat{y} = \underset{k}{\operatorname{argmax}}\, P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.4}$$

2.1.3 Support Vector Machines

Support Vector Machines a represent the training examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. When new examples are observed they are mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

2.1.4 Decision trees

The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

2.1.5 Random Forests

Bibliography